Federated LoRA Fine-Tuning with Pipelined Error-Mitigated Aggregation and Matrix-Wise Freezing

Haoran Wang[1], Xiong Wang[2], Yuqing Li[1]*, Jing Chen[1], Junyi Zhang[2], Nan Yan[1], Kun He[1], Wei Wang[3]

[1]Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University,
[2]Huazhong University of Science and Technology,
[3]Hong Kong University of Science and Technology
[1]{whr2023, li.yuqing, chenjing, nanyan, hekun}@whu.edu.cn,
[2]{xiongwang, junyizhang}@hust.edu.cn, [3]weiwa@cse.ust.hk

Abstract

Federated low-rank adaptation (LoRA) enables multiple clients to collaboratively fine-tune large language models (LLMs) without disclosing their raw data. However, existing works often experience performance degradation due to *biased* model aggregation and are hindered by *significant* communication and computation burden, both limiting training efficiency. In this paper, we propose iFLoRA, an improved Federated LoRA fine-tuning system for LLMs featuring *pipelined error-mitigated model aggregation* and *adaptive matrix-wise parameter freezing*. Specifically, iFLoRA mitigates aggregation error by first reconstructing local update matrices from clients' low-rank matrices. These are then aggregated into a global update, which is decomposed via *singular value decomposition* (SVD) to form low-rank matrices for the next round. To mitigate the overhead from SVD, iFLoRA employs a *pipeline* to overlap global aggregation, local computation, and communication. Additionally, iFLoRA implements an adaptive matrix-wise freezing scheme that assesses their *stability* and *selectively freezes* them for adaptively adjusted periods, alleviating client training overheads without compromising model performance. Extensive experiments on real-world datasets show that iFLoRA can improve time-to-target by 2.17-8.48× than state-of-the-art methods. Our code is available at: https://github.com/whr819987540/iflora.

1 Introduction

Large language models (LLMs) have revolutionized a wide range of NLP tasks, including question answering and chatbots. Fine-tuning LLMs on task-specific datasets can further enhance their applicability (Deb et al., 2025). However, traditional full-parameter fine-tuning for LLMs demands prohibitive computational and storage resources. As a prominent parameter-efficient fine-tuning (PEFT) technique, low-rank adaptation (LoRA) (Hu et al., 2022) can significantly reduce the number of trainable parameters by representing model updates as the product of two low-rank matrices.

Though efficient, task-specific data, like personal images or medical records, for LoRA fine-tuning is often dispersed across edge devices (clients), posing privacy concerns when collected (Du et al., 2023). To mitigate these risks, federated learning (McMahan et al., 2017) has been integrated with LoRA, enabling multiple clients to collaboratively fine-tune a shared LLM without centralizing their private data (Zhang et al., 2023b).

Existing federated LoRA implementations (Qiu et al., 2025; Gao et al., 2025; Sun et al., 2024; Zhang et al., 2023b) follow the standard FedAvg (McMahan et al., 2017) protocol by aggregating clients' local low-rank matrices independently. However, such aggregation can introduce a critical *error*: the product of the averaged low-rank matrices indeed *deviates* from the average of their individual products. This error is further exacerbated as the number of participants and local iterations increases and as clients' data heterogeneity intensifies, ultimately degrading model performance.

Training efficiency also poses a challenge in federated LoRA, given edge clients' limited capabilities, large model sizes and periodic exchange of LoRA modules. Empirical evidence suggests that certain modules within LLMs may *stabilize* before model convergence (Liu et al., 2021), facilitating to *selectively freeze* stabilized modules for expediting training without impacting model performance. However, existing parameter freezing approaches, from fine-grained scalar-level methods (Chen et al., 2021) to coarse-grained layer-wise schemes (Liu et al., 2021; Wang et al., 2023), are either computationally prohibitive for LLMs or risk excessive freezing and performance degradation.

In this work, we propose iFLoRA, an improved

*Corresponding author.

Federated LoRA fine-tuning system designed to enhance both model performance and training efficiency. iFLoRA achieves *error-mitigated aggregation* by first multiplying clients' respective low-rank matrices to reconstruct local update matrices, which are then aggregated and decomposed using singular value decomposition (SVD) to generate global trainable low-rank matrices for the next round. Meanwhile, LoRA rank for each update matrix is dynamically determined by exploiting the importance of singular values. This approach effectively mitigates aggregation error and ensures robust model convergence while improving training efficiency. Given the inefficiency of sequential execution and additional overhead from SVD, iFLoRA pipelines aggregation on the parameter server (PS) with clients' backward pass and parameter upload. Recognizing that different weight matrices may converge at different rates and some may only stabilize temporarily, iFLoRA implements *adaptive matrix-wise freezing* that identifies stabilized weight matrices and freezes them for adaptively adjusted periods. As a result, iFLoRA significantly reduces the communication and computational demands while preserving model accuracy.

We summarize our contributions as follows:

- We highlight the *aggregation error* and *training inefficiency* inherent in existing federated LoRA systems, and hence propose iFLoRA to overcome these challenges with both accurate model aggregation and enhanced training efficiency.
- To mitigate the aggregation error, iFLoRA reconstructs each client's local updates via low-rank matrix multiplication and proposes a SVD-based aggregation that is pipelined with clients' local computation and communication.
- iFLoRA develops adaptive matrix-wise freezing that identifies *stabilized weight matrices* and freezes them for adaptively tuned periods, reducing the communication and computational overhead without compromising model performance.
- Extensive experiments demonstrate that iFLoRA achieves *2.17-8.48*$\times$ improvement in time-to-target compared to state-of-the-art approaches.

2 Preliminaries and Motivation

2.1 Federated LoRA

As a popular PEFT method, LoRA (Hu et al., 2022) freezes the pre-trained model $W_0 \in \mathbb{R}^{m\times n}$ and adapts it with the update matrix $\Delta W \in \mathbb{R}^{m\times n}$, defined by $\Delta W = BA$. Here, $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$ are trainable low-rank matrices with rank $r \ll \min\{m, n\}$, leading to dramatic reduction in the number of trainable parameters. Regarding federated LoRA, there is a central PS and N clients $\mathcal{N} = \{1, \cdots, N\}$. Each client i holds a local model parameterized by weight matrices $W_i = \{W_{i,j,k}\}$, where j and k are indices for the Transformer layer and weight matrix type (i.e., query, key, value, or output), respectively.

Common federated LoRA implementations (Qiu et al., 2025; Gao et al., 2025; Sun et al., 2024; Zhang et al., 2023b) typically treat low-rank matrices B, A as independent parameters and employ FedAvg to aggregate them separately. In each round t, PS randomly selects a subset of clients $P^t \subset \mathcal{N}$. Upon receiving the updated low-rank matrices $\{B_i^{t+1}, A_i^{t+1}\}_{i\in P^t}$, PS performs FedAvg (McMahan et al., 2017) to build the new global trainable parameters $\bar{\omega}^{t+1} = [\hat{B}^{t+1}, \hat{A}^{t+1}]$. This process repeats until model convergence, where only lightweight trainable parameters $\{B_i^{t+1}, A_i^{t+1}\}_{i\in P^t}$, instead of full parameters $\{W_i^{t+1}\}_{i\in P^t}$, are exchanged.

However, this method is biased and incurs a critical *aggregation error*, primarily due to the incongruence between the *joint optimization* of local low-rank matrices $B_{i,j,k}^{t+1}, A_{i,j,k}^{t+1}$ performed by clients and their *separate aggregation* by PS. That is,

$$\underbrace{\bar{W}_{j,k}^{t+1} = \sum_{i\in P^t} p_i(W_{j,k}^0 + \Delta W_{i,j,k}^{t+1}) = W_{j,k}^0 + \sum_{i\in P^t} p_i B_{i,j,k}^{t+1} A_{i,j,k}^{t+1}}_{\text{Mathematically correct aggregation}}$$
$$\underbrace{\neq W_{j,k}^0 + \hat{B}_{j,k}^{t+1}\hat{A}_{j,k}^{t+1} = W_{j,k}^0 + \sum_{i\in P^t} p_i B_{i,j,k}^{t+1} \sum_{i\in P^t} p_i A_{i,j,k}^{t+1}}_{\text{Naive aggregation under federated LoRA}}, \quad (1)$$

where $P^t \subset \mathcal{N}$ is the selected clients in round t, p_i is the aggregation weight of client i with $\sum_{i\in P^t} p_i = 1$, $\bar{W}_{j,k}^{t+1}$ is the aggregated weight matrix that PS holds for weight matrix k at layer j for the t-th round, and $\hat{B}_{j,k}^{t+1}$,$\hat{A}_{j,k}^{t+1}$ denote the aggregated low-rank matrices via FedAvg.

Aggregation error remains a critical challenge (Sun et al., 2024; Singhal et al., 2025). FFA-LoRA (Sun et al., 2024) freezes low-rank matrix A and updates only B. While efficient, it often degrades model performance, as noted by FedEx-LoRA (Singhal et al., 2025), which instead appends a residual error matrix to the pre-trained weights for more accurate model aggregation. Despite comparable model performance to centralized LoRA, FedEx-LoRA incurs either heavy communication overhead of directly transmitting residual error

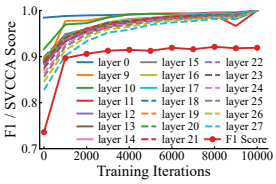


Figure 1: Convergence rates across layers.

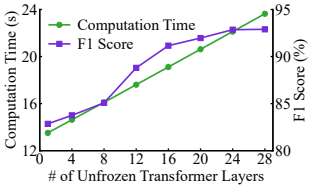


Figure 2: Computation time and model performance.

matrix or additional decomposition costs. Consequently, co-designing federated aggregation with training pipeline is essential to mitigate aggregation error while enhancing fine-tuning efficiency.

2.2 Parameter Freezing

Communication is a primary bottleneck for federated LoRA systems, due to clients' limited communication capabilities, large model sizes and periodic exchanges of LoRA updates. Recent studies indicate that certain Transformer layers tend to stabilize prior to full model convergence (Liu et al., 2021; Wang et al., 2023), suggesting the potential to reduce communication volume via parameter freezing. To elaborate, we fine-tune Llama-3.2-3B with SQuAD2.0 (Rajpurkar et al., 2018). Figure 1 reports the layers' SVCCA scores (Raghu et al., 2017) (with scores approaching 1 indicating convergence) and F1 score over training iterations, where layers with initial scores close to 1 are omitted. We observe variations in convergence rates across layers, with layers near inputs *stabilizing earlier* than those near outputs.

Despite this, the impact of parameter freezing on computational overhead remains unexplored. From Figure 2, computation time scales almost *linearly* with the number of unfrozen layers counted from the last layer, primarily due to the increased backward time. Besides, *appropriate freezing* does not significantly compromise model performance. For example, freezing the first 12 layers only decreases the F1 score from 92.89% to 91.16%, suggesting that *selectively freezing stabilized modules* can reduce parameter volume and improve system efficiency with minimal performance degradation.

However, existing works often focus on halting updates either at the fine-grained scalar level (Chen et al., 2021), which is computationally prohibitive for LLMs, or at the coarse-grained layer level (Liu et al., 2021; Wang et al., 2023), leading to excessive freezing and degraded model performance. To balance training efficiency and model performance, a more precise freezing granularity is essential.

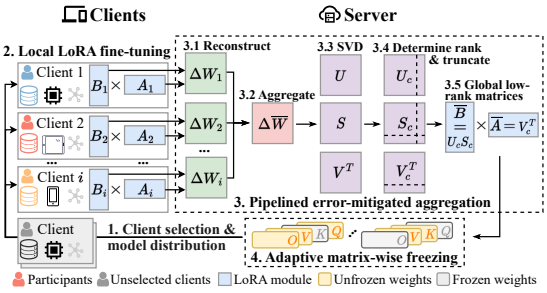


Figure 3: An overview of iFLoRA architecture.

3 iFLoRA Design

3.1 System Overview

Figure 3 depicts the main architecture of iFLoRA, with details illustrated in Algorithm 1. PS first initializes low-rank matrices $\bar{\omega}^0$, last freezing rounds FR and freezing periods FP (line 1), and then proceeds with the following steps:

① *Client selection and model distribution*. At the beginning of round t, PS randomly selects a subset of clients P^t, and broadcasts the global trainable parameters $\bar{\omega}^t$ to them (lines 3-4).

② *Local LoRA fine-tuning*. By setting $\omega_i^{t,0} = \bar{\omega}^t$, each selected client $i \in P^t$ executes E steps of SGD on its local dataset with learning rate η, i.e., $\omega_i^{t,e+1} = \omega_i^{t,e} - \eta \nabla f_i(\omega_i^{t,e}), e = 0, 1, \cdots, E-1$. It then uploads the updated low-rank matrices $\omega_{i,j,k}^{t+1} = \omega_{i,j,k}^{t,E} = [B_{i,j,k}^{t+1}, A_{i,j,k}^{t+1}]$ to PS asynchronously, without waiting for the entire model to be updated (lines 23-30).

③ *Pipelined error-mitigated aggregation with rank determination*. Upon receiving the updates from selected clients, $\{\omega_{i,j,k}^{t+1}\}_{i \in P^t}$, PS aggregates them to build the new global trainable parameters $\bar{\omega}_{j,k}^{t+1}$. Specifically, it first reconstructs each local update matrix $\Delta W_{i,j,k}^{t+1} = B_{i,j,k}^{t+1} A_{i,j,k}^{t+1}$ (3.1), and aggregates them into $\Delta \bar{W}_{j,k}^{t+1}$ (3.2), which is then decomposed into U, S, V^T via SVD (3.3). Subsequently, it determines a new LoRA rank c for $\Delta \bar{W}_{j,k}^{t+1}$ and truncates the SVD result U, S, V^T (3.4). Finally, trainable matrices $\bar{B}_{j,k}^{t+1} = U_c S_c$ and $\bar{A}_{j,k}^{t+1} = V_c^T$ are produced for the next round (3.5). Notably, pipelined aggregation proceeds at the matrix level, where PS's SVD-based aggregation (lines 5-9) overlaps with clients' backward pass and parameter upload processes (lines 28-30).

④ *Adaptive matrix-wise freezing*. At last, PS evaluates the update stability of each weight matrix, and then freezes the stabilized ones for adaptively adjusted periods (lines 10-22).

Algorithm 1: iFLoRA

Input: Clients $\mathcal{N}$, round number T, layer number L, weight matrix types $\mathcal{K}$, freezing threshold δ
Output: Federated LoRA fine-tuned parameters $\bar{\omega}^T$

// **PS**

Initialize $\bar{\omega}^0 = [\bar{B}^0, \bar{A}^0], FR, FP$
for *each round* $t \in \{0, \cdots, T-1\}$ **do**
  Select participating clients P^t from $\mathcal{N}$
  Dispatch $\bar{\omega}^t = [\bar{B}^t, \bar{A}^t]$ to all clients in P^t
  for *layer* $j \in \{L-1, \cdots, 0\}$ ***in parallel*** **do**
    for *weight matrix type* $k \in \mathcal{K}$ ***in parallel*** **do**
      Receive $\omega_{i,j,k}^{t+1}$ from client $i \in P^t$
      $B_{j,k}^{t+1} \leftarrow \{B_{i,j,k}^{t+1}\}_{i \in P^t}, A_{j,k}^{t+1} \leftarrow \{A_{i,j,k}^{t+1}\}_{i \in P^t}$
      $\Delta\bar{W}_{j,k}^{t+1}, \bar{B}_{j,k}^{t+1}, \bar{A}_{j,k}^{t+1} \leftarrow$ Run aggregation by Algorithm 2
  for *each weight matrix type* k *at each layer* j **do**
    if *matrix type* k *at layer* j *is frozen* **then**
      if $t - FR_{j,k} \geq FP_{j,k}$ **then**
        $\bar{B}^{t+1} \leftarrow \bar{B}^{t+1} \cup \bar{B}_{j,k}^{FR_{j,k}}$
        $\bar{A}^{t+1} \leftarrow \bar{A}^{t+1} \cup \bar{A}_{j,k}^{FR_{j,k}}$
    else
      Compute update score $H_{j,k}^{t+1}$ by Eq. (4)
      if $H_{j,k}^{t+1} < \delta$ **then**
        $FR_{j,k} \leftarrow t; FP_{j,k} \leftarrow FP_{j,k} + 1$
        $\bar{B}^{t+1} \leftarrow \bar{B}^{t+1} \setminus \bar{B}_{j,k}^{t+1}$
        $\bar{A}^{t+1} \leftarrow \bar{A}^{t+1} \setminus \bar{A}_{j,k}^{t+1}$
      else
        $FP_{j,k} \leftarrow \max(1, FP_{j,k}/2)$

// **Client** $i \in \mathcal{N}$

for *each round* $t \in \{0, \cdots, T-1\}$ **do**
  if $i \notin P^t$ **then**
    continue
  Receive $\bar{\omega}^t = [\bar{B}^t, \bar{A}^t]$ from PS
  Perform $E-1$ local updates
  for *layer* $j \in \{L-1, \cdots, 0\}$ ***in parallel*** **do**
    for *weight matrix type* $k \in \mathcal{K}$ ***in parallel*** **do**
      Perform E-th local update and upload $\omega_{i,j,k}^{t+1}$ to PS

3.2 Pipelined Error-Mitigated Aggregation

Existing federated LoRA systems (Qiu et al., 2025; Gao et al., 2025; Sun et al., 2024; Zhang et al., 2023b) often directly apply FedAvg to low-rank matrices, leading to notable aggregation errors, as derived in Eq. (1). To address this, we implement SVD-based error-mitigated aggregation, which is further pipelined with clients' local communication and computation processes, thereby enhancing both model performance and training efficiency.

3.2.1 Aggregation Error Mitigation

To *accurately* aggregate clients' local updates, PS multiplies clients' low-rank matrices to reconstruct their local update matrices, which are then aggregated and decomposed using SVD to yield global low-rank matrices for the next round. Throughout this process, clients only upload low-rank matrices to PS, while *compute-intensive tasks are exclusively performed on the PS*.

Decompose the aggregated update matrix. Upon receiving the updates from the selected clients, $\{[B_{i,j,k}^{t+1}, A_{i,j,k}^{t+1}]\}_{i \in P^t}$, PS first computes each client's local update matrix $\Delta W_{i,j,k}^{t+1} = B_{i,j,k}^{t+1} A_{i,j,k}^{t+1}$, and then aggregates them into the global update matrix $\Delta\bar{W}_{j,k}^{t+1}$ following the FedAvg protocol. The next step is to derive the global trainable parameters $\bar{B}_{j,k}^{t+1}, \bar{A}_{j,k}^{t+1}$ from $\Delta\bar{W}_{j,k}^{t+1}$. A straightforward approach is to factorize $\Delta\bar{W}_{j,k}^{t+1} \in \mathbb{R}^{m \times n}$ into the product of three matrices using SVD. That is,

$$SVD(\Delta\bar{W}_{j,k}^{t+1}) : \Delta\bar{W}_{j,k}^{t+1} = USV^T. \quad (2)$$

Here, $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are orthogonal matrices, and $S \in \mathbb{R}^{m \times n}$ is a diagonal matrix with non-zero singular values $\text{diag}(\sigma_1, \cdots, \sigma_r)$ in descending order, where r is the rank of global update matrix $\Delta\bar{W}_{j,k}^{t+1}$. Then, $\bar{B}_{j,k}^{t+1}$ and $\bar{A}_{j,k}^{t+1}$ can be set to $US^\alpha \in \mathbb{R}^{m \times n}$ and $S^{1-\alpha}V^T \in \mathbb{R}^{n \times n}$ for $\alpha \in \{0, 0.5, 1\}$. Our empirical results show that $\alpha = 1$ yields the superior training performance.

However, this naive setting for $\bar{B}_{j,k}^{t+1}$ and $\bar{A}_{j,k}^{t+1}$ yields *excessive zero singular values* as $r \ll \min\{m, n\}$, which unnecessarily enlarges $\bar{B}_{j,k}^{t+1}$ and $\bar{A}_{j,k}^{t+1}$, and increases the communication and computational overhead. To improve training efficiency, we *truncate* U, S, and V by selecting the first c columns of U and V, corresponding to the first c singular values in S (Hansen, 1987):

$$SVD(\Delta\bar{W}_{j,k}^{t+1}, c) : \Delta\bar{W}_{j,k}^{t+1} \approx U_c S_c V_c^T. \quad (3)$$

As a result, we obtain global trainable parameters $\bar{B}_{j,k}^{t+1} = U_c S_c \in \mathbb{R}^{m \times c}$ and $\bar{A}_{j,k}^{t+1} = V_c^T \in \mathbb{R}^{c \times n}$ for the next round. Although approximation errors are inherent in Eq. (2) and Eq. (3), experiments in §4.2 show that their magnitude is negligible, while they can effectively mitigate aggregation error.

Importance-aware rank determination. We next *identify a suitable* c for Eq. (3) to preserve model performance while enhancing training efficiency. As larger singular values essentially represent more significant model update directions, we design an *importance-aware rank determination* strategy that prioritizes the largest singular values and discards less critical ones. Specifically, we determine the

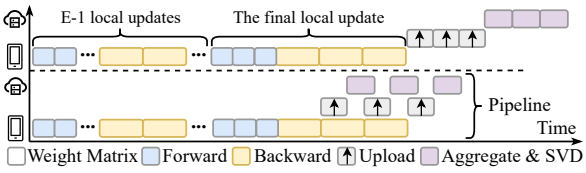


Figure 4: Pipelined aggregation of iFLoRA.

smallest c such that the squared sum of selected singular values reaches a predefined threshold θ of the total squared sum (Jolliffe, 2002). This not only offers explicit interpretability through direct control of preserved variance (Jolliffe, 2002) rather than directly thresholding singular values, but also improves empirical stability by discarding minor singular values (components associated with noise), ensuring an optimal rank-c approximation under Eckart-Young theorem (Eckart and Young, 1936).

Algorithmic details of error-mitigated aggregation are provided in Appendix B.

Remark. To address clients' heterogeneity in communication and computation, iFLoRA can be extended to *heterogeneous LoRA*. This extension employs a client-specific rank determination threshold θ_i to derive a unique LoRA rank for each client, while keeping model aggregation unchanged. Notably, the entire error-mitigated aggregation process, including compute-intensive SVD and its adaptation to heterogeneous LoRA, is performed by PS and remains *client-agnostic*.

3.2.2 Pipelined Aggregation Implenentation

The error-mitigated aggregation, based on compute-intensive SVD, can decrease training efficiency. This issue is further exacerbated by sequential execution of local training, parameter upload and global aggregation, common in existing federated LoRA systems (Qiu et al., 2025; Gao et al., 2025; Sun et al., 2024; Zhang et al., 2023b). Actually, SVD is performed on each aggregated update matrix, which inherently supports the pipelining of clients' local training, parameter upload and PS's global aggregation, as each weight matrix can be processed independently throughout these stages.

As shown in Figure 4, we present the pipelined aggregation. During the final local update, each selected client $i \in P^t$ first completes forward pass. After performing backward pass for matrix k at layer j, it immediately uploads the updated LoRA modules, $\omega_{i,j,k}^{t+1} = [B_{i,j,k}^{t+1}, A_{i,j,k}^{t+1}]$, to PS. Subsequently, the PS aggregates clients' LoRA modules $\{\omega_{i,j,k}^{t+1}\}_{i\in P^t}$ into $\Delta\bar{W}_{j,k}^{t+1}$, and then decomposes it via SVD. For each aggregated update matrix, SVD-based aggregation on the PS can be effectively overlapped with clients' backward pass and parameter upload, thus improving training efficiency. More analysis is provided in Appendix C.

Remark. We emphasize that the core motivation of this pipelined design is to hide the SVD computation overhead rather than solely reducing communication costs. Although the relative time savings of pipelining may diminish as the number of local training steps, E, increases, practical settings of federated LoRA systems typically require a small E (e.g., 5 or 10) to guarantee convergence. In such standard settings, both our theoretical analysis (Appendix C) and empirical ablation study (Section 4.5) demonstrate that the pipelined SVD is highly effective, achieving up to 10.52% and 8.98% per-round time reductions, respectively.

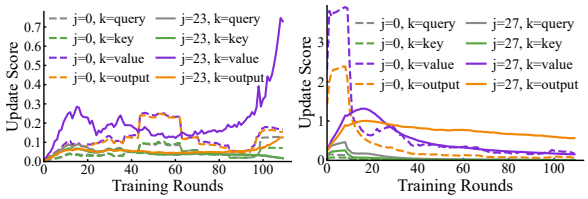
3.3 Adaptive Matrix-Wise Freezing

LoRA modules often exhibit varying convergence rates, shown in Figure 1. We develop an *adaptive matrix-wise freezing* to further improve training efficiency while preserving model performance.

Evaluate matrix update stability. Prior works have explored parameter freezing either at the fine-grained scalar level (Chen et al., 2021), which is computationally prohibitive for large-scale LLMs, or at the coarse-grained layer level (Liu et al., 2021; Wang et al., 2023), potentially leading to excessive freezing and degraded model performance. In contrast, we propose freezing at the *matrix* level, a feasible granularity that effectively balances training efficiency and model performance. Initially, we assess the update *stability* of each weight matrix $W_{j,k}^t$ by defining an update score:

$$H_{j,k}^t = \frac{1}{Q}\sum_{q=1}^{Q}\left\|\frac{\Delta\bar{W}_{j,k}^{t-q+1} - \Delta\bar{W}_{j,k}^{t-q}}{\eta}\right\|. \quad (4)$$

Here, the update differences between consecutive training rounds are averaged over Q rounds (i.e., smoothing window size) to mitigate the impact of outliers, ensuring a reliable check of update stability. A matrix with a low update score is deemed stable. Figure 5 shows the update scores across *layers* and *matrix types* when fine-tuning BERT-large and Llama-3.2-3B on the 20NEWS (Lang, 1995) and SQuAD2.0 (Rajpurkar et al., 2018) datasets, respectively. These variations among matrices of the same type and within the same layer underline the validity of matrix-wise parameter freezing.



(a) 20NEWS (BERT). (b) SQuAD2.0 (Llama-3.2).

Figure 5: Update scores across layers and matrix types.

Which matrices to freeze. While freezing specified matrices can shorten training time, it is inflexible and may degrade model performance, as shown in Figure 2. Therefore, iFLoRA adaptively decides which matrices to freeze at runtime. Recognizing that stabilized matrices contribute minimally to model performance but increase communication and computational overhead, iFLoRA freezes a matrix at the end of each round if its update score falls below the *freezing threshold* δ.

When to unfreeze. Matrix freezing is complicated by temporary stabilization of matrices, which continue to evolve after temporarily reaching steady (Chen et al., 2021). Therefore, permanent freezing can degrade model performance (Figure 2). To avoid this, iFLoRA assigns each matrix $W_{j,k}$ a *freezing period* $FP_{j,k}$. Matrix $W_{j,k}$ will be unfrozen after $FP_{j,k}$ rounds of freezing. To adjust the freezing periods, iFLoRA employs a mechanism akin to TCP's congestion control, where $FP_{j,k}$ is *additively extended* if $W_{j,k}$ remains stable, and *multiplicatively reduced* otherwise.

The whole adaptive matrix-wise freezing process is integrated into Algorithm 1. If a matrix's freezing duration surpasses the designated freezing period $FP_{j,k}$, its updates are resumed in the next round (lines 11-14). For an unfrozen matrix, its updates cease and the freezing period is additively extended if the update score is below the threshold δ (lines 16-20). Conversely, the matrices with update scores above the threshold are actively fine-tuned in the subsequent round, with their freezing periods being multiplicatively reduced (line 22).

Remark. Adaptive matrix-wise freezing is managed by the PS. It remains agnostic to the clients, and imposes no additional complexity on them compared to conventional federated LoRA systems. It only requires PS to store aggregated update matrices from previous rounds and compute the update score in Eq. (4). To make the freezing mechanism memory-efficient, PS actually stores light-weight low-rank matrices $\{[B_{i,j,k}^{t-q}, A_{i,j,k}^{t-q}]\}_{i\in P^{t-q},\forall q\in\{1,\cdots,Q\}}$, and recomputes the aggregated update matrices $\{\Delta\bar{W}_{j,k}^{t-q}\}_{\forall q\in\{1,\cdots,Q\}}$ instead of directly storing them. With this approach, PS's memory consumption decreases from 75GB to 1.03GB, with a slight time increase from 11.23ms to 68.01ms when fine-tuning Llama-3.2-3B.

Task	Dataset	# Clients	# Train	# Test	Metric	Non-IID
TC	20NEWS	100	11.3k	7.5k	Accuracy	$\beta = 0.1$
TC	AGNEWS	1000	120.0k	7.6k	Accuracy	$\beta = 0.5$
QA	SQuAD2.0	100	130.3k	11.9k	F1 score	$\times$
TS	SAMSum	100	14.7k	819	ROUGE-L	$\times$

Table 1: Statistics of datasets. TC: text classification; QA: question answering; TS: text summarization.

4 Experiments

4.1 Experimental Setup

Datasets and models. We evaluate iFLoRA on three popular LLMs and four real-world datasets. We employ Dirichlet distribution (Peng et al., 2024) $\mathcal{D} \sim \mathrm{Dir}(\beta)$ to construct clients' non-IID local datasets. Table 1 presents the statistics of datasets, across the following three types of applications:

- *Text classification*: We fine-tune BERT-large (Devlin et al., 2019) and RoBERTa-large (Liu et al., 2019) on 20NEWS (Lang, 1995) and AGNEWS (Zhang et al., 2015), and report the classification accuracy. The maximum sequence length is set to 256 and 64, respectively.
- *Question answering*: We fine-tune Llama-3.2-3B on SQuAD2.0 (Rajpurkar et al., 2018). Given that exact match and F1 scores are highly correlated, we only present the F1 score.
- *Text summarization*: We fine-tune Llama-3.2-3B on the SAMSum corpus (Gliwa et al., 2019), using ROUGE-L score to measure performance.

Baselines. We compare with following baselines:

- *FedPETuning* (Zhang et al., 2023b), as a popular federated LoRA framework, incorporates LoRA with the FedAvg training architecture.
- *FFA-LoRA* (Sun et al., 2024) freezes low-rank matrix A and updates only B. For a fair comparison, we implement it without differential privacy.
- *FedEx-LoRA* (Singhal et al., 2025) appends a residual error matrix to the pre-trained weight to achieve exact updates, and reduces communication cost with Gram-Schmidt orthogonalization.

Metrics. Following convention (Lai et al., 2021), we primarily report time-to-target, the wall-clock time required to reach a target metric (e.g., test accuracy, F1, or ROUGE-L score). For each exper-

Task	Dataset	Target	Model	FedPETuning		FFA-LoRA		FedEX-LoRA		iFLoRA	
				T2T	Speedup	T2T	Speedup	T2T	Speedup	T2T	Speedup
Text Classification	20NEWS	80.0%	BERT-large	48.46h	1.00×	98.51h	0.49×	210.95h	0.23×	**20.42h**	**2.37×**
			RoBERTa-large	89.25h	1.00×	125.69h	0.71×	118.47h	0.75×	**41.20h**	**2.17×**
	AGNEWS	90.0%	BERT-large	4.47h	1.00×	16.11h	0.28×	39.77h	0.11×	**0.96h**	**4.64×**
			RoBERTa-large	12.35h	1.00×	3.28h	3.77×	26.55h	0.47×	**1.46h**	**8.48×**
Question Answering	SQuAD2.0	91.0%	Llama-3.2-3B	46.12h	1.00×	306.69h	0.15×	174.30h	0.26×	**19.45h**	**2.37×**
Text Summarization	SAMSum	43.5%	Llama-3.2-3B	42.47h	1.00×	136.10h	0.31×	66.12h	0.64×	**19.31h**	**2.20×**

Table 2: Time-to-target (T2T) of iFLoRA and the baselines, with speedup compared to FedPETuning.

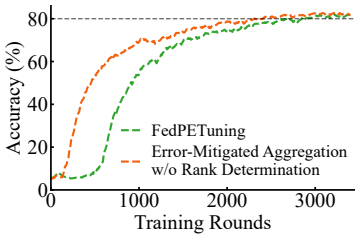


Figure 6: Round-to-target performance when fine-tuning BERT on 20NEWS.

iment, we present the average results of three runs. Besides, we detail the resource consumption, encompassing the network traffic between clients and PS, dynamic memory footprint on the client side, and overhead on the PS side in terms of SVD-based aggregation and adaptive matrix-wise freezing.

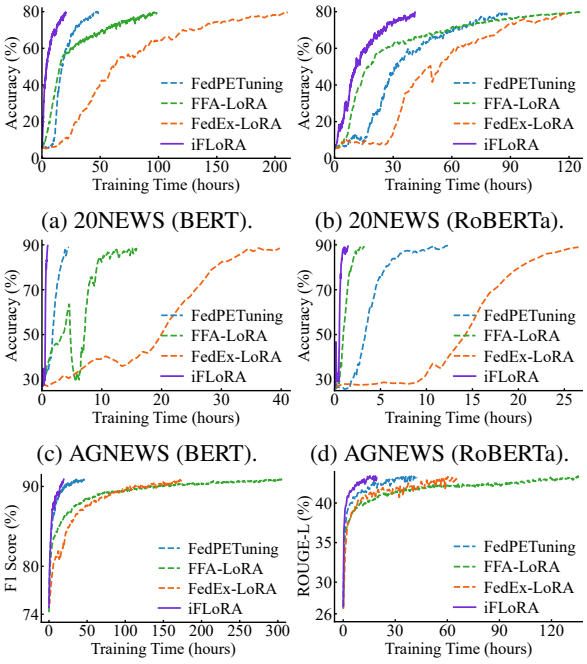
Implementation details, and hyper-parameter settings are detailed in Appendix D.1.

4.2 Mitigation of Aggregation Error

Rank determination and matrix-wise parameter freezing mechanisms in iFLoRA will alter model architecture, making round-to-target comparisons unfair for assessing our error-mitigated aggregation. Thus, prior to end-to-end evaluation, we fix the LoRA rank and disable these mechanisms to isolate the effect of error-mitigated aggregation. From Figure 6, iFLoRA surpasses FedPETuning in terms of round-to-target, demonstrating the efficacy of our SVD-based aggregation in mitigating aggregation error. While SVD in Eq. (2) inherently yields approximation error, it is negligible, averaging 10^{-10} in our experiments.

4.3 End-to-End Performance

iFLoRA improves training efficiency. Figure 7 presents the time-to-target for iFLoRA and the baselines. While FFA-LoRA outperforms FedPETuning for RoBERTa-large on AGNEWS, it shows performance degradation for other models

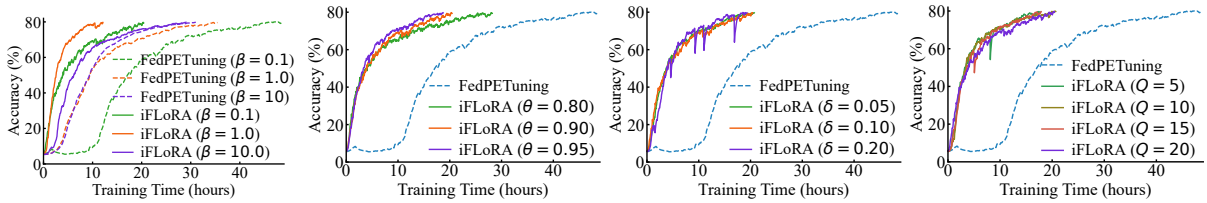


(a) 20NEWS (BERT). (b) 20NEWS (RoBERTa).
(c) AGNEWS (BERT). (d) AGNEWS (RoBERTa).
(e) SQuAD2.0 (Llama-3.2). (f) SAMSum (Llama-3.2).

Figure 7: Time-to-target performance of iFLoRA and the baselines on different datasets and LLMs.

and datasets. Even though the rank of residual error matrix is upper-bounded by $|P^t|r$ in FedEx-LoRA, its training efficiency is still hampered by the additional communication overhead of up to $(m+n)|P^t|r$. In contrast, iFLoRA consistently enhances training efficiency, suggesting its robustness across various models and datasets.

iFLoRA speeds up time-to-target. Table 2 summarizes the time-to-target and corresponding speedups of iFLoRA and the baselines. On 20NEWS, iFLoRA is 2.37× and 2.17× faster than FedPETuning for BERT-large and RoBERTa-large, respectively. Similar speedups are witnessed on AGNEWS, where iFLoRA achieves 4.64× and 8.48× speedups. For the larger Llama-3.2-3B model, iFLoRA accelerates training by 2.37× on SQuAD2.0 and 2.20× on SAMSum.



(a) Data heterogeneity. (b) Rank determination θ. (c) Freezing threshold δ. (d) Smoothing window size Q.

Figure 8: Sensitivity analysis when fine-tuning BERT-large on 20NEWS.

4.4 Sensitivity Analysis

We conduct a sensitivity analysis for iFLoRA against FedPETuning by fine-tuning BERT-large on the 20NEWS dataset.

Impact of data heterogeneity. Non-IID data across clients can lead to inconsistent model update directions, resulting in unstable and slow convergence (Shang et al., 2022; Tang et al., 2022). Typically, a smaller value of β indicates more heterogeneous local data. Figure 8a illustrates that iFLoRA consistently outperforms FedPETuning across various β. Notably, different from FedPETuning, iFLoRA *performs better under more heterogeneous settings*, specifically at $\beta = 0.1$ and $\beta = 1.0$, in contrast to $\beta = 10$. This can be attributed to our error-mitigated model aggregation.

Impact of rank determination threshold. A smaller rank determination threshold θ means fewer selected singular values and a lower LoRA rank. From Figure 8b, iFLoRA exhibits strong robustness to the changes in θ, as varying its value yield similar time-to-target results. This supports the use of a uniform, fixed ratio θ rather than directly thresholding singular values.

Impact of freezing threshold. A higher freezing threshold δ results in more weight matrices being frozen. From Figure 8c, iFLoRA is robust to δ, with comparable model performance given different values of δ. This stems from our adaptively adjusted freezing periods that can help preserve model performance.

Impact of smoothing window size. We advocate maximizing smoothing window size Q within PS's memory budget, as smaller windows amplify sensitivity to outliers from non-IID data. Figure 8d shows that iFLoRA achieves similar time-to-target speedups across varying Q. As expected, smaller Q exhibit more unstable convergence.

4.5 Ablation Study

To evaluate the effectiveness of each key component in iFLoRA, we conduct an ablation study using BERT-large on 20NEWS, as shown in Table 3. Compared with the naive approach, SVD-based error-mitigated aggregation with tightly-coupled importance-aware rank determination achieves a 2.18× speedup. Further, the pipelined aggregation expedites model training by 4.07%, effectively mitigating additional overhead from SVD and improving training efficiency. Ultimately, applying freezing alone yields a 2.12× speedup. When freezing is incorporated on top of SVD-based aggregation and pipelining, training efficiency is further improved by 4.04%, with overall speedup from 2.28× to 2.37×. This modest improvement occurs because our importance-aware rank determination already significantly reduces LoRA rank, which in turn diminishes the acceleration gains from parameter freezing.

SVD	Pipeline	Freezing	Time-to-target	Speedup
✗	✗	✗	48.46h	1.00×
✓	✗	✗	22.18h	2.18×
✓	✓	✗	21.28h	2.28×
✗	✗	✓	22.81h	2.12×
✓	✗	✓	21.05h	2.30×
✓	✓	✓	20.42h	2.37×

Table 3: Ablation study using BERT-large on 20NEWS.

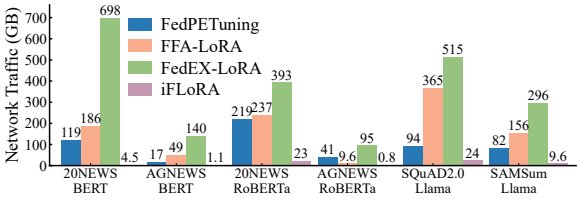
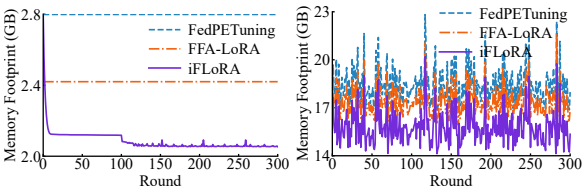


Figure 9: Network traffic.

4.6 Resource Cost

Network traffic. Figure 9 reports the network traffic, defined as the total upload and download data volume at the PS. It arises primarily from the low-rank matrices A and B transmitted during the distribution and aggregation process. Compared to FedPETuning, FFA-LoRA, and FedEx-LoRA, iFLoRA reduces the traffic by



(a) 20NEWS (BERT). (b) SAMSum (Llama-3.2).

Figure 10: Memory footprint.

74.47–97.96%, 90.19–97.81%, and 94.09–99.36%, respectively. This is achieved through two mechanisms: *matrix-wise freezing*, which halts the transmission of stabilized matrices for adaptive durations, and *rank determination*, which dynamically prunes less critical singular values to minimize matrix dimensions. Ultimately, iFLoRA significantly accelerates model convergence and lowers communication costs without compromising performance.

Memory footprint. Figure 10 shows dynamic memory footprints when fine-tuning BERT-large on 20NEWS and Llama-3.2-3B on SAMSum. Since FedEx-LoRA's footprint is identical to FedPETuning's, we report only the latter. As the sequence length varies among selected clients in each round, we report their maximum memory usage. When fine-tuning BERT-large on 20NEWS, the memory footprints of FedPETuning and FFA-LoRA remain stable as the sequence length is uniformly set to 256, while it exhibits significant fluctuations using Llama-3.2-3B that has a maximum context length of 128k. Overall, iFLoRA maintains a lower memory footprint (9.42%–25.47%) compared to FedPETuning and FFA-LoRA, thereby reducing the storage demands on the client side. Notably, the reduction in memory usage, achieved by iFLoRA's importance-aware rank determination and adaptive matrix-wise freezing, is not as significant as the reduction in network traffic. This is because memory usage of LoRA modules is substantially lower than that of the original model weights and activation values (Gao et al., 2025).

Time consumption of SVD. iFLoRA utilizes SVD to achieve error-mitigated federated aggregation. We evaluate its time consumption by running it on six NVIDIA RTX 4090 GPUs when fine-tuning Llama-3.2-3B, whose size is comparable to LLMs commonly deployed on mainstream mobile devices (Mehta et al., 2024; Team et al., 2024; Abdin et al., 2024). Though compute-intensive, SVD averages only 68.04ms per aggregated matrix. It can not only be effectively overlapped with clients' local backward pass and parameter upload processes through our pipelined aggregation, but also benefits iFLoRA by reducing clients' communication and computational overhead via importance-aware LoRA rank determination.

Memory and time consumption of freezing. In practice, PS stores light-weight low-rank matrices for freezing, instead of the aggregated update matrix in Eq. (4). With this optimization, PS's memory consumption decreases from 75GB to 1.03GB, with a slight time increase from 11.23ms to 68.01ms when fine-tuning Llama-3.2-3B.

5 Conclusion

In this paper, we present iFLoRA, an improved federated LoRA fine-tuning framework that enhances both model performance and training efficiency. iFLoRA refines the aggregation process through SVD-based federated aggregation coupled with importance-aware LoRA rank determination. By pipelining PS's aggregation with clients' backward pass and parameter upload, iFLoRA eliminates additional overhead from SVD and accelerates overall training. Moreover, iFLoRA implements an adaptive matrix-wise freezing scheme that selectively freezes stabilized weight matrices for adaptively adjusted periods, substantially reducing clients' communication and computational overhead without sacrificing model performance. Extensive experiments corroborate the superiority of iFLoRA over existing baselines in improving training efficiency while preserving model accuracy.

6 Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under grants No.62302343, 62441237, the Hubei Provincial Science and Technology Innovation Base (Platform) Program Project under grant No.2025CSA112), and the NSFC-RGC under grant No.62461160333.

Limitations

Privacy remains a critical concern within federated learning systems. While iFLoRA can be enhanced with techniques such as differential privacy to safeguard clients' privacy, its impact on model performance and training efficiency is reserved for future work. Besides, the aggregation process in iFLoRA necessitates multiplication of low-rank matrices and SVD, which is unsupported by present secure aggregation methods.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, and 1 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Chen Chen, Hong Xu, Wei Wang, Baochun Li, Bo Li, Li Chen, and Gong Zhang. 2021. Communication-efficient federated learning with adaptive parameter freezing. In *Proceedings of the 41st IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 1–11.

Yae Jee Cho, Luyang Liu, Zheng Xu, Aldi Fahrezi, and Gauri Joshi. 2024. Heterogeneous lora for federated fine-tuning of on-device foundation models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 12903–12913.

Rohan Deb, Kiran Koshy Thekumparampil, Kousha Kalantari, Gaurush Hiranandani, Shoham Sabach, and Branislav Kveton. 2025. Fishersft: Data-efficient supervised fine-tuning of language models using information gain. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, pages 12924–12943.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186.

Minxin Du, Xiang Yue, Sherman S. M. Chow, Tianhao Wang, Chenyu Huang, and Huan Sun. 2023. Dp-forward: Fine-tuning and inference on language models with differential privacy in forward pass. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2665–2679.

Carl Eckart and Gale Young. 1936. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218.

Zhidong Gao, Zhenxiao Zhang, Yuanxiong Guo, and Yanmin Gong. 2025. Federated adaptive fine-tuning of large language models with heterogeneous quantization and lora. In *Proceedings of IEEE INFOCOM 2025 - IEEE Conference on Computer Communications*, pages 1–10.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79.

Per Christian Hansen. 1987. The truncated svd as a method for regularization. *BIT Numerical Mathematics*, 27(4):534–553.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*.

Ian T Jolliffe. 2002. *Principal Component Analysis for Special Types of Data*. Springer New York, New York, NY.

Fan Lai, Yinwei Dai, Sanjay Sri Vallabh Singapuram, Jiachen Liu, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. 2022. Fedscale: Benchmarking model and system performance of federated learning at scale. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, pages 11814–11827.

Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. 2021. Oort: Efficient federated learning via guided participant selection. In *Proceedings of the 15th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 19–35.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*, pages 331–339.

Chenning Li, Xiao Zeng, Mi Zhang, and Zhichao Cao. 2022. Pyramidfl: a fine-grained client selection framework for efficient federated learning. In *Proceedings of ACM MobiCom '22: the 28th Annual International Conference on Mobile Computing and Networking*, pages 158–171.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yuhan Liu, Saurabh Agarwal, and Shivaram Venkataraman. 2021. Autofreeze: Automatically freezing model blocks to accelerate fine-tuning. *arXiv preprint arXiv:1907.11692*.

Bing Luo, Wenli Xiao, Shiqiang Wang, Jianwei Huang, and Leandros Tassiulas. 2022. Tackling system and statistical heterogeneity for federated learning with adaptive client sampling. In *Proceedings of IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, pages 1739–1748.

M-Lab. 2022. The m-lab mobiperf data set.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, Benjamin Bossan, and Marian Tietz. 2022. PEFT: State-of-the-art parameter-efficient fine-tuning methods.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-efficient learning of deep networks

from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1273–1282.

Sachin Mehta, Mohammad Sekhavat, Qingqing Cao, Max Horton, Yanzi Jin, Frank Sun, Iman Mirzadeh, Mahyar Najibikohnehshahri, Dmitry Belenko, Peter Zatloukal, and Mohammad Rastegari. 2024. Openelm: An efficient language model family with open training and inference framework. In *Proceedings of the 2nd Workshop on Efficient Systems for Foundation Models II @ ICML2024*.

NVIDIA. 2025. Nvidia jetson modules.

Zhaopeng Peng, Xiaoliang Fan, Yufan Chen, Zheng Wang, Shirui Pan, Chenglu Wen, Ruisheng Zhang, and Cheng Wang. 2024. Fedpft: Federated proxy fine-tuning of foundation models. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4806–4814.

Wenqi Qiu, Yipeng Zhou, Jinzhi Wang, Quan Z. Sheng, and Laizhong Cui. 2025. Flm-topk: Expediting federated large language model tuning by sparsifying intervalized gradients. In *Proceedings of IEEE INFOCOM 2025 - IEEE Conference on Computer Communications*, pages 1–10.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. 2017. SVCCA: singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 6076–6085.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 784–789.

Pengjie Ren, Chengshun Shi, Shiguang Wu, Mengqi Zhang, Zhaochun Ren, Maarten de Rijke, Zhumin Chen, and Jiahuan Pei. 2024. Melora: Mini-ensemble low-rank adapters for parameter-efficient fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3052–3064.

Xinyi Shang, Yang Lu, Gang Huang, and Hanzi Wang. 2022. Federated learning on heterogeneous and long-tailed data via classifier re-training with federated features. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2218–2224.

Raghav Singhal, Kaustubh Ponkshe, and Praneeth Vepakomma. 2025. Fedex-lora: Exact aggregation for federated and efficient fine-tuning of large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1316–1336.

Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. 2024. Improving lora in privacy-preserving federated learning. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*.

Zhenheng Tang, Yonggang Zhang, Shaohuai Shi, Xin He, Bo Han, and Xiaowen Chu. 2022. Virtual homogeneity learning: Defending against data heterogeneity in federated learning. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, pages 21111–21132.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Madeleine Udell and Alex Townsend. 2019. Why are big data matrices approximately low rank? *SIAM Journal on Mathematics of Data Science*, 1(1):144–160.

Yiding Wang, Decang Sun, Kai Chen, Fan Lai, and Mosharaf Chowdhury. 2023. Egeria: Efficient DNN training with knowledge-guided layer freezing. In *Proceedings of the 18th European Conference on Computer Systems (EuroSys)*, pages 851–866.

Ziyao Wang, Zheyu Shen, Yexiao He, Guoheng Sun, Hongyi Wang, Lingjuan Lyu, and Ang Li. 2024. Flora: Federated fine-tuning large language models with heterogeneous low-rank adaptations. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 22513–22533.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023a. Adaptive budget allocation for parameter-efficient fine-tuning. In *Proceedings of the 11th International Conference on Learning Representations*.

Ruiyi Zhang, Rushi Qiang, Sai Ashish Somayajula, and Pengtao Xie. 2024. Autolora: Automatically tuning matrix ranks in low-rank adaptation based on meta learning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 5048–5060.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, pages 649–657.

Zhuo Zhang, Yuanhang Yang, Yong Dai, Qifan Wang, Yue Yu, Lizhen Qu, and Zenglin Xu. 2023b. Fedpetuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models. In *Findings of the Association for Computational Linguistics (ACL)*, pages 9963–9977.

A Related Work

Centralized LoRA fine-tuning. LoRA has been recognized as an effective PEFT techniques for LLMs (Hu et al., 2022). AdaLoRA (Zhang et al., 2023a) parameterizes model updates in the form of SVD result and adjusts LoRA rank by quantifying the sensitivity of parameters to the training loss. MELoRA (Ren et al., 2024) develops mini-ensemble low-rank adapters that achieve a higher rank and better performance with fewer parameters fine-tuned. AutoLoRA (Zhang et al., 2024) uses meta learning to automatically determine the optimal rank for each LoRA layer. However, these centralized LoRA fine-tuning methods are not directly applicable in federated LoRA systems, primarily due to performance degradation caused by client heterogeneity.

Federated LoRA fine-tuning. With increasing privacy concerns, federated learning (McMahan et al., 2017) has emerged as a promising solution that enables clients to collaboratively fine-tune LLMs without disclosing their raw data (Qiu et al., 2025). FedPETuning (Zhang et al., 2023b) proposes a federated parameter-efficient fine-tuning framework using FedAvg. HETLORA (Cho et al., 2024) develops heterogeneous LoRA via zero-padding and truncation. FAH-QLoRA (Gao et al., 2025) quantizes the base model and assigns heterogeneous LoRA ranks across clients. Though effective, these approaches suffer from the aggregation error and heavy communication overheads. As a concurrent work to our study, FLoRA (Wang et al., 2024) achieves noise-free federated aggregation in heterogeneous LoRA by stacking local LoRA modules, but it continues to face high communication and computational overheads. FFA-LoRA (Sun et al., 2024) freezes randomly-initialized low-rank matrix A and only updates zero-initialized matrix B. Although FFA-LoRA can eliminate the aggregation error and improve training efficiency, it degrades model performance. FedEx-LoRA (Singhal et al., 2025) appends a residual error matrix to pre-trained weight matrix to mitigate the aggregation error. Despite comparable model performance with centralized LoRA, FedEx-LoRA introduces either heavy communication overhead if directly transmitting the full-size residual error matrix, or additional decomposition overhead.

Parameter freezing. Parameter freezing is widely used to reduce communication and computation overheads (Sun et al., 2024). APF (Chen et al., 2021) identifies stable scalars and halts their updates. AutoFreeze (Liu et al., 2021) freezes Transformer layers that are close to convergence. Egeria (Wang et al., 2023) designs knowledge-guided layer freezing for DNN training. However, these methods focus on halting updates either at the fine-grained scalar level, which is computationally prohibitive for LLMs, or at the coarse-grained layer level, which may lead to excessive freezing and performance loss.

Algorithm 2: Error-Mitigated Aggregation

Input: Selected clients P^t, layer j, matrix type k, low-rank matrices $B_{j,k}^{t+1}$, $A_{j,k}^{t+1}$, rank determination threshold θ

Output: Aggregated update matrix $\Delta\bar{W}_{j,k}^{t+1}$ and global low-rank matrices $\bar{B}_{j,k}^{t+1}$, $\bar{A}_{j,k}^{t+1}$

1 **for** *each client* $i \in P^t$ **do**
2 $\quad \Delta W_{i,j,k}^{t+1} \leftarrow B_{i,j,k}^{t+1} A_{i,j,k}^{t+1}$
3 $\Delta\bar{W}_{j,k}^{t+1} \leftarrow \sum_{i \in P^t} p_i \Delta W_{i,j,k}^{t+1}$
4 $U, S, V^T \leftarrow SVD(\Delta\bar{W}_{j,k}^{t+1})$
5 $c_{j,k}^{t+1} \leftarrow 0; S_{cum} \leftarrow 0; S_{total} \leftarrow \sum_{\sigma \in S} \sigma^2$
6 **for** *each singular value* $\sigma \in S$ **do**
7 $\quad S_{cum} \leftarrow S_{cum} + \sigma^2$
8 $\quad c_{j,k}^{t+1} \leftarrow c_{j,k}^{t+1} + 1$
9 $\quad$ **if** $S_{cum} \geq \theta \times S_{total}$ **then**
10 $\quad\quad$ **break**
11 $\bar{B}_{j,k}^{t+1} \leftarrow U_{c_{j,k}^{t+1}} S_{c_{j,k}^{t+1}}; \bar{A}_{j,k}^{t+1} \leftarrow V_{c_{j,k}^{t+1}}^T$
12 **return** $\Delta\bar{W}_{j,k}^{t+1}, \bar{B}_{j,k}^{t+1}, \bar{A}_{j,k}^{t+1}$

B Error-Mitigated Aggregation

Algorithm 2 illustrates the SVD-based error-mitigated aggregation process performed on PS. Here, we assume that client i holds a private dataset $\mathcal{D}_i$ containing $D_i = |\mathcal{D}_i|$ data samples, and p_i is the aggregation weight of client i with $\sum_{i \in \mathcal{N}} p_i = 1$, often $p_i = \frac{D_i}{\sum_{i \in \mathcal{N}} D_i}$. PS first reconstructs each local update matrix (lines 1-2), computes the aggregated update matrix $\Delta\bar{W}_{j,k}^{t+1}$ (line 3) and then decomposes it using Eq. (2) (line 4). After that, the minimum number $c_{j,k}^{t+1}$ of singular values required to reach the threshold θ (lines 5-10) is determined as the LoRA rank for matrix truncation in Eq. (3). Finally, PS produces the new global low-rank matrices from the truncated SVD results (line 11).

C Pipelined Aggregation

We now theoretically analyze how pipelined aggregation can reduce per-round training time. For simplicity, we focus specifically on computations within the self-attention layer. For any weight matrix across L Transformer layers, let $\mathbf{T}_i^F$ denote the

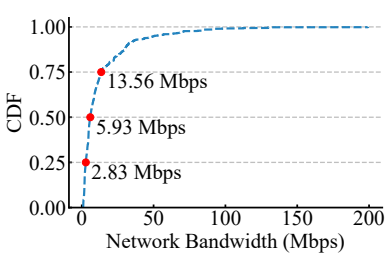


Figure 11: Clients' network bandwidth.

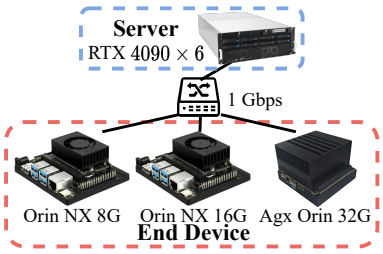


Figure 12: Network topology of physical devices.

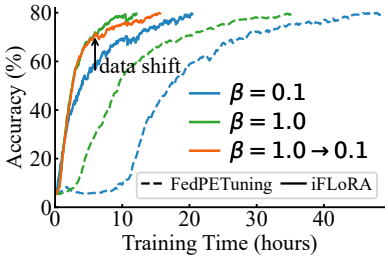


Figure 13: Data distribution shift.

average forward pass time for client i, while the average times for the backward pass and parameter upload are represented as $\mathbf{T}_i^B$ and $\mathbf{T}_i^C$, respectively. Accordingly, the average time for SVD-based aggregation is denoted by $\mathbf{T}^A$. Then, the per-round time for sequential execution of these processes can be roughly expressed as:

$$\max_{i\in P^t}\left\{|\mathcal{K}|EL(\mathbf{T}_i^F+\mathbf{T}_i^B)+|\mathcal{K}|L\mathbf{T}_i^C\right\}+|\mathcal{K}|L\mathbf{T}^A, \tag{5}$$

where E is the number of local updates and $\mathcal{K}$ is the set of weight matrix types. Similarly, we can derive the pipelined execution time, i.e.,

$$\max_{i\in P^t}\left\{|\mathcal{K}|EL(\mathbf{T}_i^F+\mathbf{T}_i^B)+\mathbf{T}_i^C\right\}+\mathbf{T}^A. \tag{6}$$

By profiling each component, we estimate the time reduction achieved from pipelined aggregation. In particular, we fine-tune Llama-3.2-3B on SQuAD2.0 (Rajpurkar et al., 2018) with $E = 5$, $L = 28$, $\mathcal{K} = \{\text{query}, \text{key}, \text{value}, \text{output}\}$, and mini-batch size as 2. Utilizing network bandwidth sampled from real-world datasets (Lai et al., 2022; M-Lab, 2022), the average times for clients' forward pass, backward pass and parameter upload on NVIDIA Jetson AGX Orin (NVIDIA, 2025) are 64.49ms, 119.83ms and 36.91ms, respectively. The average SVD-based aggregation time on the PS equipped with six NVIDIA RTX 4090 GPUs, is 72.55ms. From Eq. (5) and Eq. (6), pipelined training can reduce the per-round time from 115.48s to 103.33s, yielding a 10.52% reduction.

Remark. Hardware heterogeneity among clients may raise potential concerns, as it directly affects the time required for local backward pass and parameter upload. Consequently, the PS may receive clients' updated LoRA modules at staggered times, thereby prolonging the aggregation and impeding the pipeline efficiency. While pipelined training can be adversely affected by extreme stragglers, this issue, also present in sequential execution, can be effectively mitigated by redundant client sampling (Lai et al., 2021), strategic client selection (Luo et al., 2022; Li et al., 2022) and heterogeneous LoRA ranks (Gao et al., 2025).

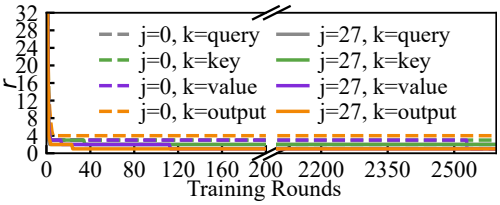


Figure 14: Rank r across layers and matrices over training rounds under importance-aware rank determination.

D Experiments

D.1 Experimental Setup

Implementation details. We conduct experiments on a GPU server equipped with 256GB RAM and six NVIDIA RTX 4090 GPUs, each with 24GB of VRAM, to emulate the clients. To ensure the authenticity of results, clients' upload and download bandwidths are sampled from a real-world dataset (Lai et al., 2022), as shown in Figure 11. Meanwhile, computation time is profiled by running on NVIDIA Jetson AGX Orin and Orin NX (NVIDIA, 2025), which possess resources comparable to mainstream mobile devices. Figure 12 shows the network topology of our hardware devices. Consistent with previous studies (Wang et al., 2024; Cho et al., 2024; Zhang et al., 2023b), we apply LoRA to the self-attention layer in each Transformer block, namely the query, key, value and output projection matrices. We utilize HuggingFace PEFT library (Mangrulkar et al., 2022) to fine-tune LLMs with LoRA.

Hyper-parameters. In each round, PS randomly selects 10 clients as participants. The mini-batch size is set to 4 for text classification task and 2 for other tasks, with local step $E = 5$ and learning rate $\eta = 0.01$. The initial LoRA rank is 32. Besides, the

rank determination threshold θ and freezing threshold δ are 0.9 and 0.1, respectively. On NVIDIA RTX 4090 GPUs, we set the smoothing window size Q to 20 by default.

D.2 Robustness against Data Distribution Shift

We next explore the robustness of iFLoRA under time-varying heterogeneous data distributions. To simulate this, we adjust the concentration parameter β from 1 to 0.1 when fine-tuning BERT-large on 20NEWS. As illustrated in Figure 13, the training curve corresponding to the data distribution shift closely resembles that of the static data distribution case. This confirms that iFLoRA remains robust when adapting to data distribution shift.

D.3 Robustness of SVD Threshold

Since the number of singular values c may differ across layers and matrices due to their varying distributions, is *a uniform, fixed threshold θ to all LLM layers and matrices* still efficient and robust? Figure 14 reports the adjusted rank r over training rounds during the fine-tuning of Llama-3.2-3B on SQuAD2.0 (Rajpurkar et al., 2018). We observe that our importance-aware rank determination exhibits excellent stability in r. This is because, though the distributions of singular values may differ significantly across layers and matrices, they are generally highly concentrated (Udell and Townsend, 2019), with large singular values dominating the total squared sum. Hence, it's sufficient to capture these large singular values with a uniform, fixed ratio. Overall, the selection of the most critical c singular values establishes an adaptive LoRA rank for the aggregated update matrix $\Delta\bar{W}_{j,k}^{t+1}$, effectively preserving model performance while enhancing training efficiency.